

# On the Generation of L-Convex Polyominoes

Paolo Massazza\*  
paolo.massazza@uninsubria.it

## Abstract

We present a simple but efficient method for generating the set  $\text{LPol}(n)$  of L-convex polyominoes of size  $n$ . We show a bijection between  $\text{LPol}(n)$  and a suitable set of pairs of integer sequences. This lets us design a CAT (Constant Amortized Time) algorithm for generating  $\text{LPol}(n)$  using  $O(\sqrt{n})$  space.

## 1 Introduction

A polyomino  $P$  is a finite connected set of edge-to-edge adjacent square unit cells in the cartesian two-dimensional plane, defined up to translations. Several classes of polyominoes have been considered and extensively studied in enumerative [4, 12] or bijective combinatorics [3, 11, 18]. They have been studied in two-dimensional language theory and image treatment [8, 9], in particular from the point of view of the problem of reconstruction of  $P$  from partial informations [1, 14]. They have also deserved a particular attention in tiling theory, where they describe tile shapes [13, 17].

Here we are interested in the class of L-convex polyominoes, introduced in [7] and later studied from a combinatorial point of view in [5, 6], where the authors enumerate them with respect to the semi-perimeter and to the area. In particular, we study the problem of designing an efficient algorithm for the exhaustive generation of L-convex polyominoes of a given size  $n$ .

We recall that a CAT (constant amortized time) algorithm for the exhaustive generation of parallelogram polyominoes of size  $n$  has been recently proposed in [15], where it is shown how to exploit a bijection between the set of parallelogram polyominoes of size  $n$  and a set of suitable pairs of integer partitions. By adopting a similar approach, we provide a bijection between L-convex polyominoes and a suitable set of integer sequences, and we show that such a bijection directly leads to a CAT algorithm for generating the set of all L-convex polyominoes of size  $n$ .

## 2 Preliminaries

A polyomino  $P$  is called *convex* if the intersection of  $P$  with an infinite row or column of connected cells is always connected. We indicate by  $\text{CPol}(n)$  the set

---

\*Università degli Studi dell'Insubria, Dipartimento di Scienze Teoriche e Applicate - Sezione Informatica, Via Mazzini 5, 21100 Varese, Italy

of convex polyominoes of size  $n$ . The *size* of  $P$  is the number of its cells,  $s(P)$ . The height and the width of  $P$ , denoted by  $h(P)$  and  $w(P)$ , respectively, are the height and the width of the minimal rectangle which bounds  $P$ , that is,

$$\begin{aligned} h(P) &= \max\{|j_1 - j_2| : \exists i_1, i_2, (i_1, j_1), (i_2, j_2) \in P\} + 1, \\ w(P) &= \max\{|i_1 - i_2| : \exists j_1, j_2, (i_1, j_1), (i_2, j_2) \in P\} + 1. \end{aligned}$$

A *path* in  $P$  is a sequence  $c_1, c_2, \dots, c_k$  of cells of  $P$  such that for all  $i$ , with  $1 \leq i < k$ ,  $c_i$  shares one edge with  $c_{i+1}$ . Convex polyominoes can be classified according to the number of vertices they share with minimal bounding rectangles. In particular,  $P$  can be a *Ferrer diagram* if it shares three vertices with the minimal bounding rectangle, while it is a *stack polyomino* if it shares at least two adjacent vertices. A *rectangular polyomino* or *rectangle* is a convex polyomino of height  $x$ , width  $y$  and size  $xy$ .

Ferrer diagrams of size  $n$  can be represented as integer partitions of  $n$ . An *integer partition* of  $n$  (also called linear partition) is a non-increasing sequence of nonnegative integers with sum  $n$ . We indicate by  $\text{LP}(n)$  the set of integer partitions of  $n$ . For any two integers  $x, p$  with  $p > 0$ , we denote by  $x^{[p]}$  the sequence  $\underbrace{(x, \dots, x)}_p$  and by  $\cdot$  the *catenation product* of sequences. The *length* of

$t = (t_1, \dots, t_l) \in \text{LP}(n)$  is  $l(t) = l$ , the *height* is  $h(t) = t_1$  and the *size* (or *weight*) is  $s(t) = \sum t_i = n$ . See that  $t$  can be univocally written as  $t = t_1^{[m_1]} \cdot t_2^{[m_2]} \cdot \dots \cdot t_{j_k}^{[m_k]}$  with  $t_1 > t_{j_2} > \dots > t_{j_k}$  and  $m_i > 0$ . The following notations are useful when dealing with integer sequences,  $t_{<i} = (t_1, \dots, t_{i-1})$  ( $t_{\leq i} = (t_1, \dots, t_i)$ ) and  $t_{>i} = (t_{i+1}, \dots, t_l)$  ( $t_{\geq i} = (t_i, \dots, t_l)$ ). The set  $\text{LP}(n)$  can be ordered with respect to the *negative lexicographic* order defined as follows: let  $t = (t_1, \dots, t_m)$  and  $v = (v_1, \dots, v_l)$  belong to  $\text{LP}(n)$ , we set  $t <_{\text{nlex}} v$  if and only if

$$\exists i, 1 \leq i \leq \min(l, m) : t_j = v_j, 1 \leq j < i, t_i > v_i.$$

Analogously, a stack polyomino of size  $n$  can be represented by a *unimodal sequence* of size  $n$ , that is, an integer sequence  $t_1, \dots, t_l$  with  $\sum_i t_i = n$  and such that  $t_1 \leq t_2 \leq \dots \leq t_i \geq t_{i+1} \geq \dots \geq t_l$ . We denote by  $\text{US}(n)$  the set of unimodal sequences of size  $n$ . A unimodal sequence  $t$  is univocally written as  $t = t_{<i} \cdot t_i^{[h]} \cdot t_{>i+h}$ , where  $t_{<i}$  and the reversal of  $t_{>i+h}$  are integer partitions of height at most  $t_i - 1$ .

**Definition 1** Given two unimodal sequences of size  $n$ ,  $t = t_{<i} \cdot t_i^{[h]} \cdot t_{>i+h}$  and  $t' = t'_{<j} \cdot t'_j{}^{[k]} \cdot t'_{>j+k}$ , we set  $t < t'$  if and only if  $t_i > t'_j$  or  $t_i = t'_j$  and  $h > k$  or  $t_i = t'_j, h = k$  and  $s(t_{<i}) > s(t'_{<j})$  or  $t_i = t'_j, h = k, s(t_{<i}) = s(t'_{<j})$  and  $t_{<i} <_{\text{nlex}} t'_{<j}$  or  $t_i = t'_j, h = k, t_{<i} = t'_{<j}$  and  $t_{>i+h} <_{\text{nlex}} t'_{>j+k}$ .

The *vertical projection* of  $P$  is the integer vector  $\pi(P) = (\pi_1, \dots, \pi_l)$  where  $l = w(P)$  and for all  $i$ , with  $1 \leq i \leq l$ ,  $\pi_i = \#\{j | (i, j) \in P\}$ . Moreover, the *position vector* of  $P$  is defined as the integer vector  $\sigma(P) = (\sigma_1, \dots, \sigma_l)$  where  $\sigma_i = \min\{j | (i, j) \in P\}$ , with  $1 \leq i \leq l$ .

Given an integer vector  $v$ , with  $\sum_i v_i = n$ , and a class of polyominoes  $A \subseteq \text{CPol}(n)$ , we consider the set  $\Pi_A(v) = \{P \in A | \pi(P) = v\}$ . Any  $P \in A$  is univocally described by a pair of integer sequences  $(v, p)$  where  $v = \pi(P)$  and  $p = \sigma(P)$ . We say that a vector  $p \in \mathbb{Z}^l$  is *A-feasible* with  $v \in \mathbb{N}^l$  if and only if  $(v, p)$  identifies a polyomino  $P \in A$ .

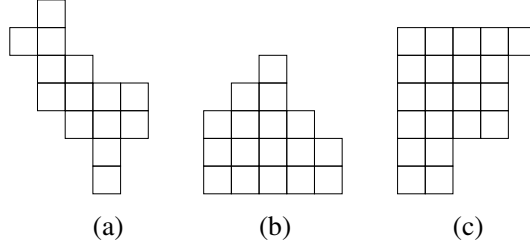
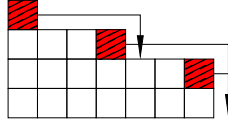


Figure 1: A convex polyomino (a), a stack polyomino (b), a Ferrer diagram (c).

## 2.1 Integer partitions with constraints

We consider the set  $LP^h(n) \subseteq LP(n)$  of the integer partitions of  $n$  into parts smaller or equal to  $h$ . The smallest linear partition of  $LP^h(n)$  is  $t = h^{[a]} \cdot b$  where  $a = \lfloor n/h \rfloor$  and  $b = \langle n \rangle_h$ , with  $l(t) = \lceil n/h \rceil$ . In the sequel we are interested in generating  $LP^h(n)$  by means of an efficient algorithm. We show that the same approach used in [15] can be used to design a CAT (Constant Amortized Time) algorithm which produces the ordered sequence (w.r.t  $<_{nlex}$ ) of elements in  $LP^h(n)$ . In order to do that, we define a discrete dynamical system (with parameter  $h$ ) whose states are all and only the elements of  $LP^h(n)$ . This system has an evolution rule called *Move* which simulates the movements of  $n$  grains stacked into a (two-dimensional) silo of height  $h$  and width  $n$ . Grains move to the right only. Informally, the rightmost grain of a plateau, say in column  $i$ , falls down to the first column  $j$ , with  $j > i$ , such that its height differs by at least 2.



More formally, one has:

**Definition 2** Given  $t \in LP^h(n)$  and an integer  $i$ , with  $1 \leq i \leq l(t)$ , let  $k$  be the first integer greater than  $i$  such that  $t_i - t_k \geq 2$  and  $t_i - t_j = 1$  for all  $j$  with  $i < j < k$  ( $k$  is undefined,  $k = \perp$ , if no such integer exists). Then

$$Move(t, i) = \begin{cases} t_{<i} \cdot (t_i - 1, t_{i+1}, \dots, t_{k-1}, t_k + 1) \cdot t_{>k} & \text{if } k \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

We write  $t \xrightarrow{i} t'$  if  $t' = Move(t, i)$  and, more generally,  $t \xrightarrow{*} v$  if there is a sequence of moves leading from  $t$  to  $v$ . We denote by  $G(t) = \{v | t \xrightarrow{*} v\}$  the set of states which can be reached from  $t \in LP^h(n)$ . In particular, by reasoning as in [15, Lemma 3] one can easily prove:

**Lemma 1** Let  $t = \min_{<_{nlex}}(LP^h(n))$ . Then, one has  $G(t) = LP^h(n)$ .

We denote by  $Rmost(t)$  the index indicating the rightmost move in  $t$ ,  $Rmost(t) = \max\{i | Move(t, i) \neq \perp\}$ . The following definition is of particular interest for the exhaustive generation of  $LP^h(n)$ .

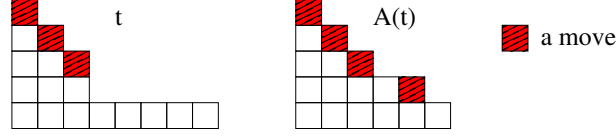


Figure 2: The grand ancestor of  $t = (5, 4, 3, 1, 1, 1, 1) \in \text{LP}^6(17)$ .

**Definition 3** Let  $t \in \text{LP}^h(n)$  and  $i = \text{Rmost}(t)$ . The grand ancestor of  $t$  is the integer partition  $A(t) = \min_{<_{\text{nlex}}} \{v \in \text{LP}^h(n) | v_{\leq i} = t_{\leq i}, \text{Move}(v, i) \neq \perp\}$ .

From this definition it immediately follows:

**Lemma 2** Let  $t \in \text{LP}^h(n)$ ,  $i = \text{Rmost}(t)$  and  $q = \sum_{j>i} t_j$ . Then,  $A(t) = t_{\leq i} \cdot (t_i - 1)^{[a]} \cdot b$ , where  $a = \lfloor \frac{q}{t_i - 1} \rfloor$  and  $b = \langle q \rangle_{t_i - 1}$ .

When  $n$  and  $h$  are clear by the context, we denote by  $t^{(e)}$  the  $e$ th integer partition in  $\text{LP}^h(n)$ . The following lemma states how to generate the ordered sequence of elements in  $\text{LP}^h(n)$ .

**Lemma 3** Let  $t \in \text{LP}^h(n)$ . Then, for any  $e > 0$ , we have

$$A(t^{(e)}) \stackrel{i_e}{\Rightarrow} t^{(e+1)}, \quad \text{with } i_e = \text{Rmost}(t^{(e)}).$$

*Proof.* We follow the proof of [16, Lemma 2.16], where a similar result applies to the set of states of a discrete dynamical system associated with the Ice Pile Model.

Let  $A(t^{(e)}) \stackrel{i_e}{\Rightarrow} v$ . By Lemma 2 and Definition 2 it follows that  $v = t_{\leq i_e}^{(e)} \cdot (t_{i_e}^{(e)} - 1)^{[a+1]} \cdot (b+1)$  for suitable integers  $a$  and  $b$ . Suppose  $t^{(e+1)} <_{\text{nlex}} v$  and note that  $t_{\leq i_e}^{(e+1)} = t_{\leq i_e}^{(e)}$ . We distinguish two cases. If  $t_{i_e}^{(e+1)} = t_{i_e}^{(e)} - 1 = v_{i_e}$  one necessarily has  $t_j^{(e+1)} = t_{i_e}^{(e)} - 1 = v_j$ , with  $i_e < j \leq i_e + a$ , and  $t_{i_e+a+1}^{(e+1)} > b+1 = v_{i_e+a+1}$ , that is,  $t^{(e+1)} \in \text{LP}^h(m)$ , with  $m > n$ .

Otherwise, one has  $t_{\leq i_e}^{(e+1)} = t_{\leq i_e}^{(e)}$  and  $t_{>i_e}^{(e)} <_{\text{nlex}} t_{>i_e}^{(e+1)}$ . Since  $i_e = \text{Rmost}(t^{(e)})$ , the sequence  $t_{>i_e}^{(e)}$  is equal to  $1^{[p]}$ , for a suitable  $p \geq 0$ . This means that  $t_{>i_e}^{(e)} \not<_{\text{nlex}} t_{>i_e}^{(e+1)}$  (otherwise  $t_{>i_e}^{(e+1)} = 1^{[q]}$  with  $q < p$  and  $t^{(e+1)}$  would not be in  $\text{LP}^h(n)$ ). ■

As a matter of fact,  $\text{LP}^h(n)$  turns out to be a lattice with respect to the relation induced by  $\Rightarrow$ . The top and the bottom are the two integer partitions  $h^{[a]} \cdot b$  and  $1^{[n]}$ , respectively. The previous lemma lets us design a function  $\text{Next} : \text{LP}^h(n) \mapsto \text{LP}^h(n) \cup \{\perp\}$  such that  $\text{Next}(t^{(e)}) = t^{(e+1)}$  for all  $e > 0$  (we define  $\text{Next}(t^{(e)}) = \perp$  if  $\text{Rmost}(t^{(e)}) = \perp$ , that is, if  $t^{(e)} = 1^{[n]}$ ). Such a function implicitly defines a spanning tree associated with the lattice  $\text{LP}^h(n)$ , that is, a tree with radix  $t^{(1)} = h^{[a]} \cdot b$  and such that for any  $e > 1$  the father of  $t^{(e)}$  is  $A(t^{(e-1)})$ , see Figure 3. Moreover,  $\text{Next}$  can be implemented so that Algorithm 1 is CAT. More precisely, one has:

**Theorem 1**  $\text{INTPARTGEN}(n, h)$  generates  $\text{LP}^h(n)$  in time  $O(|\text{LP}^h(n)|)$  using  $O(\sqrt{n})$  space.

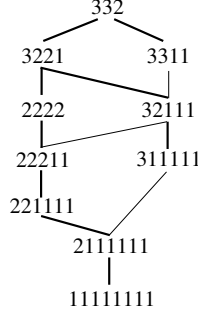


Figure 3: The lattice  $LP^3(8)$  and its spanning tree (thick edges).

*Proof (outline).* The space complexity follows from Section 3.1. With respect to the time complexity, the result follows by noting that, for any  $t \in LP^h(n)$ , Lemma 2 lets us compute  $A(t)$  in time  $O(1)$ . Finally, it is immediate to see that  $\text{Move}(A(t), i)$  runs in time  $O(1)$  and that  $\text{Rmost}(t)$  can be updated (after a move) in time  $O(1)$  too. ■

---

**Algorithm 1** Exhaustive Generation of  $LP^h(n)$ .

---

```

PROCEDURE INTPARTGEN( $n, h$ )
 $t := \text{INIT}(n, h)$ ;  $\{t \leftarrow \min_{<_{\text{nlex}}}(LP^h(n)); \text{Rmost}(t) \leftarrow 1;\}$ 
while  $\text{Rmost}(t) \neq \perp$  do
     $t := \text{NEXT}(t)$ ;  $\{t \leftarrow A(t); t \leftarrow \text{Move}(t, i); \text{UPDATE}(\text{Rmost}(t));\}$ 
end while

```

---

## 2.2 L-Convex Polyominoes

A convex polyomino  $P$  is said L-convex if any two cells of  $P$  are connected by a path in  $P$  with at most one change of direction. We indicate by  $L\text{Pol}(n)$  the set of L-convex polyominoes of size  $n$ . The following theorem provides a characterization of  $L\text{Pol}(n)$  in terms of suitable pairs of integer sequences.

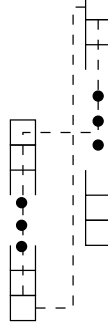
**Theorem 2** *Let  $v \in \mathbb{N}^l$  and  $p \in \mathbb{Z}^l$ . Then  $(v, p)$  individuates a polyomino  $P \in L\text{Pol}(n)$ , with  $\pi(P) = v$  and  $\sigma(P) = p$ , if and only if*

1.  $v \in US(n)$ ;
2. for all  $i, j$  with  $1 \leq i < j \leq l$ , either  $p_j \leq p_i < p_i + v_i - 1 \leq p_j + v_j - 1$  or  $p_i \leq p_j < p_j + v_j - 1 \leq p_i + v_i - 1$ .

*Proof.*  $(\Rightarrow)$  Let  $P \in L\text{Pol}(n)$  and consider the pair  $(v, p)$  with  $v = \pi(P)$ ,  $p = \sigma(P)$ . First, suppose that  $v$  is not unimodal. Then there are  $i, j$  with  $i < j$  such that  $v_i > v_{i+1} = \dots = v_{j-1} < v_j$ . This implies either that  $P$  is not convex or that it is not L-convex (there is not an L-path either from the upper cell in column  $i$  to the lower cell in column  $j$  or from the lower cell in column  $i$  to the upper cell in column  $j$ ).

Now, let  $\bar{i}, \bar{j}$  be two integers such that property 2 does not hold. This means that either  $p_{\bar{i}} < p_{\bar{j}} \leq p_{\bar{i}} + v_{\bar{i}} - 1 < p_{\bar{j}} + v_{\bar{j}} - 1$  or  $p_{\bar{j}} < p_{\bar{i}} \leq p_{\bar{i}} + v_{\bar{i}} - 1 < p_{\bar{j}} + v_{\bar{j}} - 1$ .

In the first case, as shown below in the picture, there is not an L-path from the lower cell in column  $\bar{i}$  to the upper cell in column  $\bar{j}$  of  $P$ . Similarly, in the second case no L-path exists in  $P$  from the upper cell in column  $\bar{i}$  to the lower cell in column  $\bar{j}$ .



( $\Leftarrow$ ) Properties 1 and 2 imply that the pair  $(v, p)$  individuates a polyomino  $P \in \text{CPol}(n)$ . Now, suppose that  $P$  is not L-convex and let  $i_1, i_2$  be two integers, with  $i_1 < i_2$ , such that there is not an L-path from  $(i_1, j_1) \in P$  to  $(i_2, j_2) \in P$ . Since  $v$  is unimodal, if  $v_{i_1} \leq v_{i_2}$  then for all  $k$  with  $i_1 \leq k \leq i_2$  one has  $v_{i_1} \leq v_k$ . So, by property 2, we get  $p_k \leq p_{i_1} \leq p_{i_1} + v_{i_1} - 1 \leq p_k + v_k - 1$  and there is an L-path from any cell in column  $i_1$  of  $P$  to any cell in column  $i_2$ . More precisely, the L-path from  $(i_1, j_1)$  to  $(i_2, j_2)$  consists of  $i_2 - i_1$  East-steps followed by  $|j_2 - j_1|$  South-steps (North-steps) if  $j_1 > j_2$  ( $j_1 \leq j_2$ ). Analogously, if  $v_{i_1} \geq v_{i_2}$  then for all  $k$  with  $i_1 \leq k \leq i_2$  one has  $v_k \geq v_{i_2}$ , and then  $p_k \leq p_{i_2} \leq p_{i_2} + v_{i_2} - 1 \leq p_k + v_k - 1$ . Thus, there is an L-path from  $(i_1, j_1)$  to  $(i_2, j_2)$ : one starts with  $|j_2 - j_1|$  South-steps (North-steps) if  $j_1 > j_2$  ( $j_1 \leq j_2$ ), followed by  $i_2 - i_1$  East-steps. ■

In order to generate  $\text{LPol}(n)$ , we consider the following order relation.

**Definition 4** Let  $P, P' \in \text{LPol}(n)$  with  $(v, p) = (\pi(P), \sigma(P))$  and  $(v', p') = (\pi(P'), \sigma(P'))$ . Then  $P < P'$  if and only if  $v < v'$  or  $v = v'$  and  $p <_{\text{lex}} p'$ .

### 3 The Problem

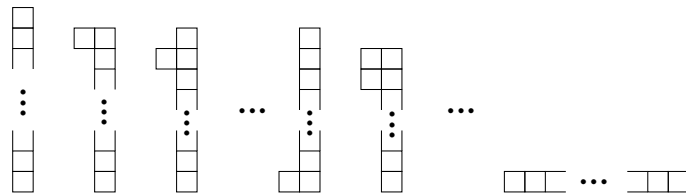
Here we are interested in the following problem:

**Problem** L-Convex Polyominoes Generation (LCPG)

**Input** an integer  $n$ ;

**Output** the ordered sequence of L-convex polyominoes in  $\text{LPol}(n)$ .

Our approach is that of producing the sequence



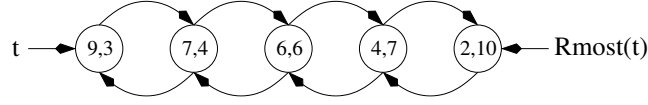


Figure 4: The representation of  $t = 9^{[3]} \cdot 7 \cdot 6^{[2]} \cdot 4 \cdot 2^{[3]} \in \text{LP}^{10}(56)$ . Since  $t$  has only 5 different values, the list has 5 nodes. Each node contains an integer and an index indicating the rightmost position where it occurs in  $t$ .

by generating the sequence of pairs  $(\pi(P), \sigma(P))$  characterized in Theorem 2. A direct application of Theorem 2 and Definition 4 lets us develop a generation algorithm. This consists of an iterator for unimodal sequences equipped with a procedure for generating the set of integer sequences which are LPol-feasible with a given vertical projection.

### 3.1 The Data Structure

In order to deal with an L-convex polyomino  $P$  given by  $(\pi(P), \sigma(P))$ , we choose a suitable representation for both components. First, the unimodal sequence  $v = \pi(P) = v_{<i} \cdot v_i^{[a]} \cdot v_{>i+a}$  is split into three components,  $F = v_{<i}$ ,  $R = v_i^{[a]}$  and  $F' = v_{>i+a}$ , where  $F, F'$  are Ferrer diagrams (integer partitions of height at most  $v_i - 1$ ) and  $R$  is a rectangle (of size  $a \times v_i$ ). We represent integer partitions by doubly linked lists (see Figure 4). So, given,  $t \in \text{LP}^h(n)$  let  $I = \{i | t_i > t_{i+1}\} = \{i_1, \dots, i_d\}$ , with  $i_1 < i_2 < \dots < i_d$ . Then, the list representing  $t$  has as many elements as different values in  $t$ , that is,  $d = \#I$  different nodes. For all  $j$  with  $1 \leq j \leq d$ , the  $j$ th node contains two integers, the value  $t_{i_j}$  and the largest index  $k_j$  such that  $t_{k_j} = t_{i_j}$ ,

$$(t_1, k_1), (t_{j_2}, k_2), \dots, (t_{j_d}, k_d), \approx t_1^{[k_1]} \cdot t_{j_2}^{[k_2 - k_1]} \cdot \dots \cdot t_{j_d}^{[k_d - k_1 - \dots - k_{d-1}]} = t.$$

Now, let us consider  $p = \sigma(P)$ . By Property 2 in Theorem 2, one has that  $v_k = v_{k+1}$  implies  $p_k = p_{k+1}$ . Thus, since  $\pi(P) = v_{<i} \cdot v_i^{[a]} \cdot v_{>i+a}$ , we have a list for  $p_{<i}$  (with the same number of nodes of the list for  $v_{<i}$ ), a list for  $p_{>i+a}$  (with the same number of nodes of the list for  $v_{>i+a}$ ) and one integer  $b$  which indicates the position of  $R$ ,  $p_i = p_{i+1} = \dots = p_{i+a} = b$ . Without loss of generality, from here on we always suppose  $b = 1$ .

Lastly, the rightmost move in  $t$  is just a link (either to the last node of the list representing  $t$  -if  $t_{l(t)} > 1$ - or to the second last node -if  $t_{l(t)} = 1$ ), while the rectangle  $R$  is simply coded by the integer pair  $(v_i, a)$ .

### 3.2 The algorithm

We provide here an algorithm for the exhaustive generation of  $\text{LPol}(n)$ . Since an L-convex polyomino  $P$  is represented by the pair  $(\pi(P), \sigma(P))$ , the idea is that of defining an iterator for  $\text{US}(n)$  which, for each generated unimodal sequence  $u$ , say of length  $l$ , computes also the set of all  $p \in \mathbb{Z}^l$  which are LPol-feasible with  $u$ . Actually, the iterator for  $\text{US}(n)$  consists of one iterator for rectangles (RECT) and two iterators for Ferrer diagrams (LFERRER, RFERRER). Indeed, one can generate  $\pi(P) = v_{<i} \cdot v_i^{[a]} \cdot v_{>i+a}$  as follows: first set  $v_i$  and  $a$  (this is done

by RECT), then generate the reversal of  $v_{<i}$  (the integer partition represented by the list  $t$  in LFERRER), and finally compute  $v_{>i+a}$  (represented by the list  $t'$  in RFERRER). Once  $\pi(P)$  is known, Procedure POSITION is called. Since  $\sigma(P)$  is of the form  $p_{<i} \cdot 1^{[a]} \cdot p_{>i+a}$ , the procedure can easily exploit Theorem 2 in order to set the values in the nodes of the lists  $p, p'$  representing  $p_{<i}$  and  $p_{>i+a}$ , respectively. Indeed, the procedure works as follows. It receives as inputs the links to the two highest columns (in  $v_{<i}$  and in  $v_{>i+a}$ ) with unknown position, together with the height  $h$  and the position  $pos$  of the latest (and higher) processed column (initially  $v_i$  and 1, respectively). Then, the position of the higher column (possibly a group of adjacent columns having the same height, say  $k$ ) gets (iteratively) all the values such that Property 2 in Theorem 2 is satisfied. These are integers in the range  $[pos, h - k]$ .

---

**Algorithm 2** Iterator for rectangles.

---

```

PROCEDURE RECT( $n$ )
  for  $h:=n$  downto 2 do
    for  $w:=\lfloor n/h \rfloor$  downto 1 do
      R.height :=  $h$ ; R.width :=  $w$ ;
       $m:=n-hw$ ;
      LFERRER( $m, h-1$ );
    end for
  end for
  R.height := 1; R.width :=  $n$ ;

```

---



---

**Algorithm 3** Iterator for the left Ferrer diagram.

---

```

PROCEDURE LFERRER( $n, h$ )
  for  $m:=n$  downto 1 do
     $t:=\text{INIT}(m, h)$ ;  $\{t \leftarrow \min_{<\text{nlex}}(\text{LP}^h(m)); \text{Rmost}(t) \leftarrow 1;\}$ 
    RFERRER( $n-m, h$ );
    while  $\text{Rmost}(t) \neq \perp$  do
       $t:=\text{NEXT}(t)$ ;  $\{t \leftarrow A(t); t \leftarrow \text{Move}(t, i); \text{UPDATE}(\text{Rmost}(t));\}$ 
      RFERRER( $n-m, h$ );
    end while
  end for
  RFERRER( $n, h$ );

```

---



---

**Algorithm 4** Iterator for the right Ferrer diagram.

---

```

PROCEDURE RFERRER( $n, h$ )
   $t':=\text{INIT}(n, h)$ ;  $\{t' \leftarrow \min_{<\text{nlex}}(\text{LP}^h(n)); \text{Rmost}(t') \leftarrow 1;\}$ 
  POSITION( $t, t', h+1, 1$ );
  while  $\text{Rmost}(t') \neq \perp$  do
     $t':=\text{NEXT}(t')$ ;  $\{t' \leftarrow A(t'); t' \leftarrow \text{Move}(t', i); \text{UPDATE}(\text{Rmost}(t'));\}$ 
    POSITION( $t, t', h+1, 1$ );
  end while

```

---

Figure 5 illustrates part of the tree associated with the call POSITION( $t, t', h, 1$ ), with  $t$  representing the integer partition  $4^{[2]} \cdot 2^{[2]}$ ,  $t'$  representing  $3 \cdot 2^{[2]} \cdot 1$  and



---

**Algorithm 5** Computation of the position vector.

---

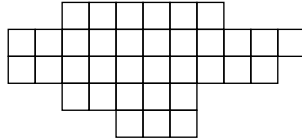
```

PROCEDURE POSITION( $t, t', h, \text{pos}$ )
  if  $t \rightarrow \text{height} > t' \rightarrow \text{height}$  then
    for  $m := h - t \rightarrow \text{height}$  downto 0 do
       $p \rightarrow \text{pos} := \text{pos} + m$ ;  $p = p \rightarrow \text{next}$ ;
      POSITION( $t \rightarrow \text{next}, t', t \rightarrow \text{height}, \text{pos} + m$ );
    end for
  else
    if  $t' \rightarrow \text{height} > t \rightarrow \text{height}$  then
      for  $m := h - t' \rightarrow \text{height}$  downto 0 do
         $p' \rightarrow \text{pos} := \text{pos} + m$ ;  $p' = p' \rightarrow \text{next}$ ;
        POSITION( $t, t' \rightarrow \text{next}, t' \rightarrow \text{height}, \text{pos} + m$ );
      end for
    else  $\{t' \rightarrow \text{height} = t \rightarrow \text{height}\}$ 
      for  $m := h - t' \rightarrow \text{height}$  downto 0 do
         $p \rightarrow \text{pos} := \text{pos} + m$ ;  $p' \rightarrow \text{pos} := \text{pos} + m$ ;
         $p = p \rightarrow \text{next}$ ;  $p' = p' \rightarrow \text{next}$ ;
        POSITION( $t \rightarrow \text{next}, t' \rightarrow \text{next}, t' \rightarrow \text{height}, \text{pos} + m$ );
      end for
    end if
  end if
end if

```

---

$h = 5$ . This call occurs during the generation of polyominoes in LPol(35) with  $\pi(P) = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 3 \cdot 2^{[2]} \cdot 1$ . Each node corresponds to a recursive call and is labelled with either three or four entries which identify the column (or a group of adjacent columns having the same height) being processed: the name(s) of the list(s) to which the column belongs to, the height and the position. Note that the height of the tree is at most the sum of the lengths of the two lists  $t, t'$ . Moreover, for any  $i > 0$ , an internal node at level  $i$  refers to a column (or a group of adjacent columns) which is higher than any column associated with a node at level  $i + 1$ . Obviously, there is a one-to-one correspondence between the set of leaves of the tree and the set of polyominoes in LPol(35) having vertical projection  $2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 3 \cdot 2^{[2]} \cdot 1$ . For example, the third leaf (from the left) corresponds to the following L-convex polyomino (with position vector  $3^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 3^{[3]} \cdot 4$ )



### 3.3 Complexity

For any  $P \in \text{LPol}(n)$ , the space required to represent  $(\pi(P), \sigma(P))$  is clearly  $O(\sqrt{n})$ , as shown in Section 3.1. With respect to the time complexity, we first consider Procedure POSITION for which one has the following:

**Lemma 4** *Let  $\gamma = \alpha \cdot k^{[a]} \cdot \beta \in \text{US}(n)$ , with  $h(\alpha), h(\beta) < k$ . Then POSITION( $t, t', k, 1$ ) (with  $t$  representing the reversal of  $\alpha$  and  $t'$  representing  $\beta$ ) runs in time  $O(|\Pi_{\text{LPol}}(\gamma)|)$ .*

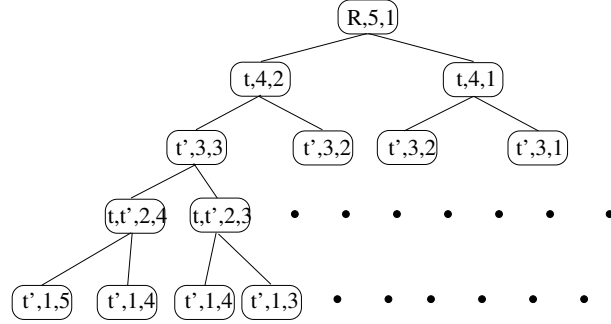


Figure 5: The execution tree of POSITION.

*Proof (outline).* We consider the execution tree  $T_\gamma$ , like the one shown in Figure 5, and let  $f(\gamma)$  and  $g(\gamma)$  be the number of leaves and the number of nodes of  $T_\gamma$ , respectively. Observe that

- $f(\gamma) = |\Pi_{\text{LPol}}(\gamma)|$ ;
- $g(\gamma) = O(f(\gamma))$ . Indeed, every internal node has at least two children, since there are at least two valid positions for the column(s) associated with them;
- if  $v$  is a node with  $r$  children, then the  $r$  different positions associated with the children are computed in time  $O(r)$ .

Therefore, the execution of  $\text{POSITION}(t, t', k, 1)$  requires time  $O(|\Pi_{\text{LPol}}(\gamma)|)$ . ■

Lastly, as a direct application of the previous lemma, one has:

**Theorem 3** *There is a CAT algorithm which solves Problem LCPG.*

*Proof.* By Theorem 1, it follows that procedures RECT, LFERRER and RFERRER let us generate  $\text{US}(n)$  in constant amortized time. Then, since Procedure POSITION is called on each unimodal sequence, the result follows from Lemma 4. ■

### 3.4 Conclusions

The method we have presented lets us generate all L-convex polyominoes of size  $n$  in constant amortized time. Obviously, also stack polyominoes can be generated by a CAT algorithm, since they are coded by unimodal sequences. Thus, it is quite natural to ask whether a similar approach can be used to design CAT algorithms for other interesting subclasses of polyominoes. Indeed, efficient generation algorithms for convex polyominoes and for column convex polyominoes were presented in [10]. Such algorithms use the ECO method [2] for generating polyominoes with respect to the semi-perimeter. Nevertheless, we think that the approach here presented could also be used to design CAT generation algorithms (with respect to the size) for at least these two last classes of polyominoes.

## References

- [1] Barcucci, E., Del Lungo, A., Nivat, M., Pinzani, R., Reconstructing convex polyominoes from horizontal and vertical projections. *Theoret. Comput. Sci.*, 155 n.2 (1996), 321–347.
- [2] Barcucci, E., Del Lungo, E., Pergola, E., Pinzani, R., ECO: a methodology for the Enumeration of Combinatorial Objects, *J. of Diff. Eq. and App.*, 5 (1999), 435–490.
- [3] Barcucci, E., Frosini, A., Rinaldi, S., Direct-convex polyominoes: ECO method and bijective results. *Proc. of Formal Power Series and Algebraic Combinatorics 2002*, R.Brak, O. Foda, C. Greenhill, T. Guttman, A. Owczarek (Eds.), Melbourne (2002).
- [4] Bousquet-Mélou, M., A method for the enumeration of various classes of column-convex polygons, *Discrete Math.*, 154 (1996), 1–25.
- [5] Castiglione, G., Frosini, A., Munarini, E., Restivo, A., Rinaldi, S., Combinatorial aspects of L-convex polyominoes, *Eur. J. Comb.*, 28 n.6 (2007), 1724–1741.
- [6] Castiglione, G., Frosini, A., Restivo, A., Rinaldi, S., Enumeration of L-convex polyominoes by rows and columns, *Theoret. Comput. Sci.*, 347 n.1-2 (2005), 336–352.
- [7] Castiglione, G., Restivo, A., Reconstruction of L-convex Polyominoes, *Electron. Notes in Discrete Math.*, 12 Elsevier Science (2003), 290–301.
- [8] Castiglione, G., Vaglica, R., Recognizable Picture Languages and Polyominoes, *Proc. of the 2nd International Conference on Algebraic Informatics CAI 2007*, LNCS 4728 (2007), 160–171.
- [9] De Carli, F., Frosini, A., Rinaldi, S., Vuillon, L., On the Tiling System Recognizability of Various Classes of Convex Polyominoes, *Ann. Comb.*, 13 (2009), 169–191.
- [10] Del Lungo, A., Duchi, E., Frosini, A., Rinaldi, S., On the generation and enumeration of some classes of convex polyominoes, *Electron. J. Combin.*, 11 (2004), #R60.
- [11] Del Lungo, A., Mirolli, M., Pinzani, R., Rinaldi, S., A Bijection for Directed-Convex Polyominoes, *Proc. of DM-CCG 2001*, Discrete Mathematics and Theoretical Computer Science AA (2001), 133–144.
- [12] Delest, M., Viennot, X.G., Algebraic languages and polyominoes enumeration, *Theoret. Comput. Sci.*, 34 (1984), 169–206.
- [13] Golomb, S. W., Checker Boards and Polyominoes, *The American Mathematical Monthly*, 61 (1954), 675–682.
- [14] Kuba, A., Balogh, E., Reconstruction of convex 2D discrete sets in polynomial time, *Theoret. Comput. Sci.*, 283 n.1 (2002), 223–242.

- [15] Mantaci, R., Massazza, P., From Linear Partitions to Parallelogram Polyominoes, *Proc. of DLT 2011*, LNCS 6975 (2011), 350–361.
- [16] Massazza, P., Radicioni, R., A CAT algorithm for the exhaustive generation of ice piles, *RAIRO Theoretical Informatics and Applications*, 44 n.4 (2010), 525–543.
- [17] Ollinger, N., Tiling the Plane with a Fixed Number of Polyominoes, *Proc. of LATA 2009*, LNCS 5457 (2009), 638–647.
- [18] Pergola, E., Sulanke, R.A., Schröder Triangles, Paths, and Parallelogram Polyominoes, *Journal of Integer Sequences*, 1 (1998), Article 98.1.7.